(12) UK Patent Application (19) GB (11) 2 238 636 (13) A

(43) Date of A publication 05.06.1991

(21) Application No 9016596.0

(22) Date of filing 27.07.1990

(30) Priority data
(31) 444663 (32) 01.12.1989 (33) US

(71) Applicant
Sun Microsystems Inc

(Incorporated in the USA - Delaware)

2550 Garcia Avenue, Mountain View, California 94043, United States of America

(72) Inventor
David S.H. Rosenthal

(74) Agent and/or Address for Service
Potts Kerr and Co
15 Hamilton Square, Birkenhead, Merseyside, L41 6BR, United Kingdom

(51) INT CL[5]
G06F 1/00 12/14

(52) UK CL (Edition K)
G4A AAP

(56) Documents cited
GB 2181281 A EP 0254854 A2
Micro-Systemes, Nov '86, No 69, (France), pp 94-97, P. Forme, "PC/NOS: a distinctive local network"

(58) Field of search
UK CL (Edition K) G4A AAP
INT CL[5] G06F 1/00 12/14
Online databases: WPI, INSPEC

(54) X window security system

(57) The method of rendering an X Windows server system running on a server and at least one host computer terminal secure includes allowing users to view only resources of the X Windows server system the use of which has been specifically authorized to that user, and allowing users to manipulate only resources of the X Windows server system the use of which has been specifically authorized to that user.

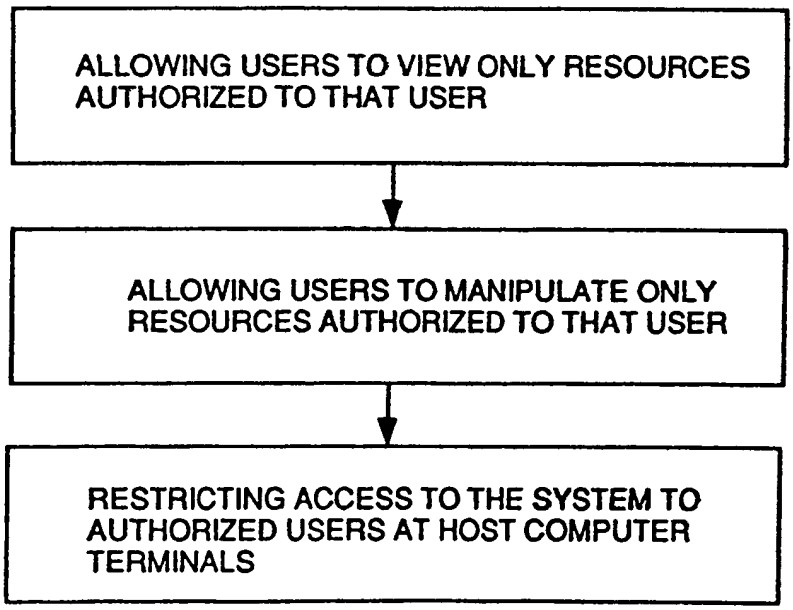

FIGURE 2

GB 2 238 636

The specification as filed includes a computer program which is not reproduced here; it may be inspected in accordance

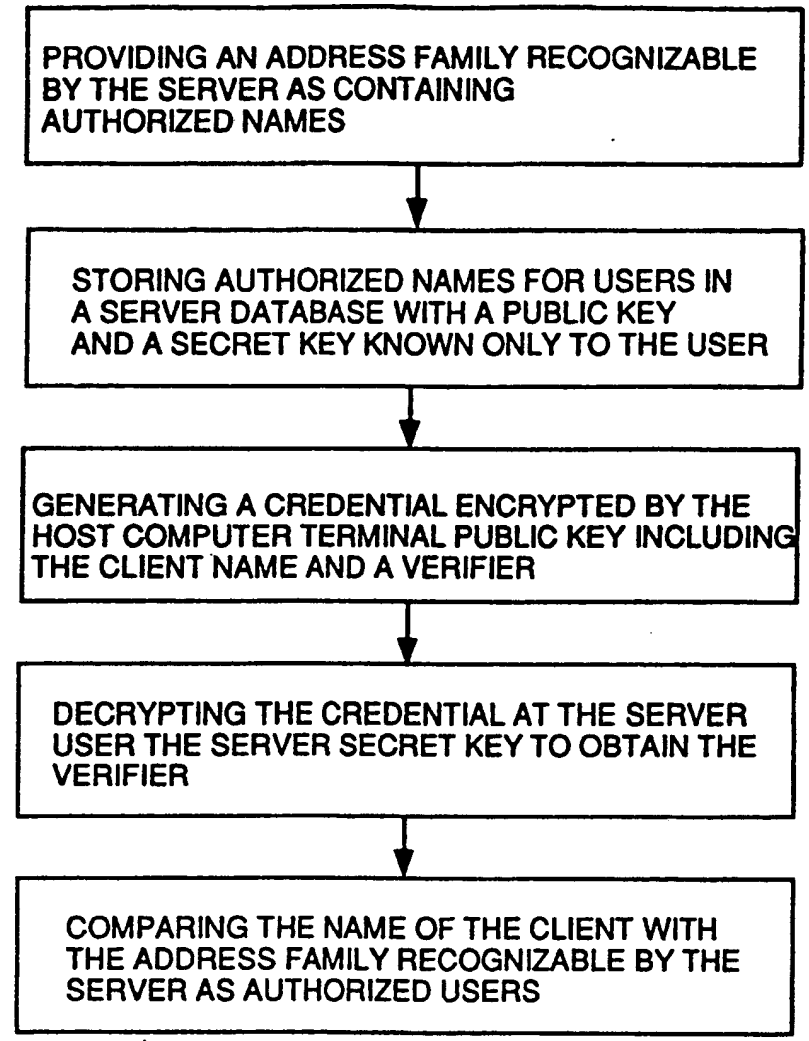

FIGURE 1

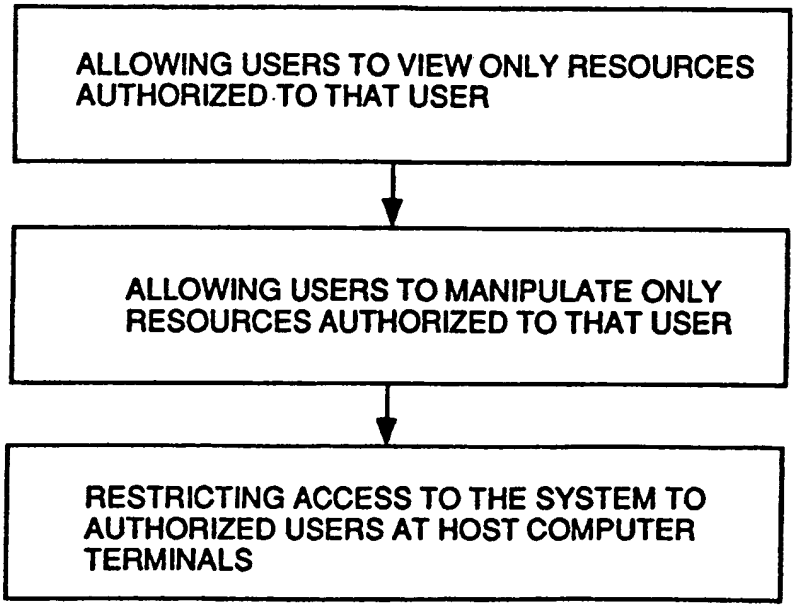

FIGURE 2

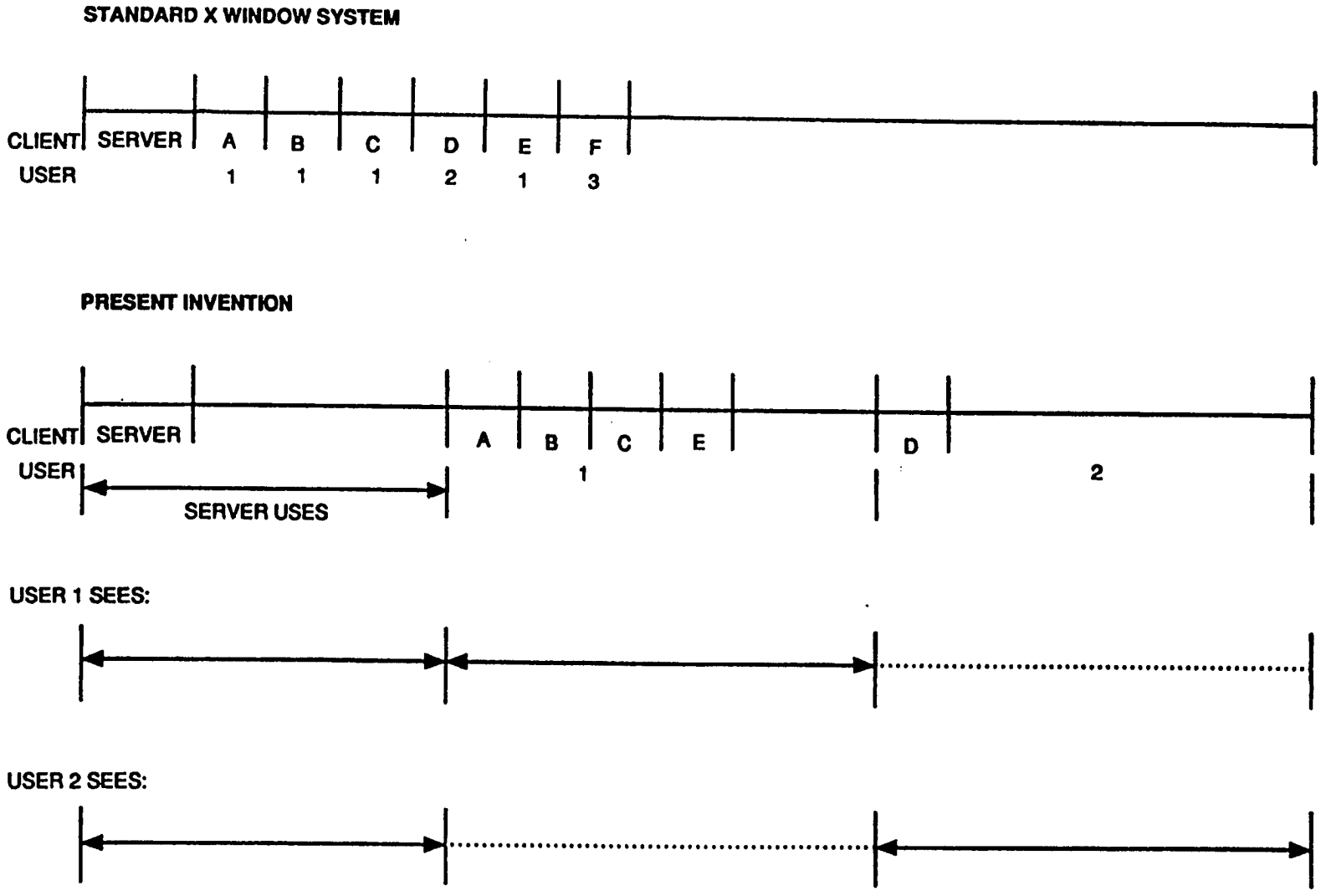
STANDARD X WINDOW SYSTEM
CLIENT
SERVER
A
B
C
D
E
F
USER
1
1
1
2
1
3
PRESENT INVENTION
CLIENT
SERVER
A
B
C
E
D
USER
1
2
SERVER USES
USER 1 SEES:
USER 2 SEES:

FIGURE 3

X WINDOW SECURITY SYSTEM.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates to networked window systems and, more particularly, to a method and apparatus for rendering the X Window System more secure.

2. History of the Prior Art

Arranging individual computer work stations in a network is desirable for a number of reasons. Such an arrangement allows a number of different individuals to work together on the same piece of work. It allows the sharing of high cost computer assets such as printers, character readers, central memory and other resources which might be under-utilized by individuals operating individual work stations. Windowing systems, on the other hand, are desirable for both individual work stations and networks because they allow users to run a number of processes essentially simultaneously whether by time sharing a single central processing unit or by utilizing a plurality of processors to implement individual processes. Window arrangements provide, in effect, a plurality of displays appearing together or separately on an individual computer output display, each such display or window displaying one of a number of what may be simultaneously operating applications.

There have been a number of networking systems devised with windowing capabilities. One of the most recent such systems is the X Window System developed by Schiefler and Gettys at the Massachusetts Institute of Technology as a system for sharing university computer resources. The X Window System has become quite popular because it can be utilized with

different types of computers and it allows users to tailor interfaces to the particular application or network involved. The X Window System is, in many respects, like an operating system which runs on a centralized server, providing its facilities to all of the users networked to that server.

One problem with the X Window System is that it is insecure in at least two respects. There is no control over who may access the resources of the system nor over what they may do with the resources once access has been gained. The gaining of access is referred to herein as "authorization." The restricting of operations by an "authorized" user is referred to herein as "access control." Although it attempts to limit gaining access to authorized users, the X Window System does so by allowing only certain computers called hosts, the names for which are held in the server, to access that server. Any intruder who has gained access to any of these computers then has unrestricted access to the server. For example, the intruder may observe a user's key strokes, button pushes, and mouse motions; and monitor any output visible on the display. Such an intruder can disrupt the normal operation of the applications by the user in any of a number of ways, for example, by destroying the user's windows.

The X Window System was designed for use in a university environment in which the overhead of sophisticated resource protection mechanisms was considered inappropriate. However, the X Window System now finds itself used in situations where industrial and governmental espionage are serious considerations and where the pernicious snooping and interference with computer programs is well-known. This expanded use of the X Window system

highlights the need of that system for arrangements to make the system more secure.

SUMMARY OF THE INVENTION

It is, therefore, an object of this invention to limit the persons authorized to use the X Window System.

It is another object of the present invention to limit the resources available to authorized users of the X Window system.

It is an additional and more particular object of the present invention to limit the resources available to authorized users of the X Window system to selected resources without indicating to such users that additional resources are available to others.

These and other objects of the present invention are realized in the present invention by a method of rendering an X Windows server system running on a server and at least one host computer terminal secure comprising the steps of allowing users to view only resources of the X Windows server system the use of which has been specifically authorized to that user, and allowing users to manipulate only resources of the X Windows server system the use of which has been specifically authorized to that user or to a server manager.

These and other objects and features of the present invention will be better understood by reference to the detailed description of the invention which follows taken together with the drawings in which like elements are referred to by like designations throughout the several figures.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a flow chart describing the operation of the invention in broad detail.

Figure 2 is a flow chart describing the operation of the invention in more particular detail.

Figure 3 is a diagram illustrating the allocation of resources in an X Window System in accordance with the present invention.

NOTATION AND NOMENCLATURE

Some portions of the detailed descriptions which follow are presented in terms of algorithms and symbolic representations of operations on data bits within a computer memory. These algorithmic descriptions and representations are the means used by those skilled in the data processing arts to most effectively convey the substance of their work to others skilled in the art.

An algorithm is here, and generally, conceived to be a self-consistent sequence of steps leading to a desired result. The steps are those requiring physical manipulations of physical quantities. Usually, though not necessarily, these quantities take the form of electrical or magnetic signals capable of being stored, transferred, combined, compared, and otherwise manipulated. It has proven convenient at times, principally for reasons of common usage, to refer to these signals as bits, values, elements, symbols, characters, terms, numbers, or the like. It should be borne in mind, however, that all of these and similar terms are to be associated with the appropriate physical quantities and are merely convenient labels applied to these quantities.

Further, the manipulations performed are often referred to in terms, such as adding or comparing, which are commonly associated with mental operations performed by a human operator. No such capability of a human operator is necessary or desirable in most cases in any of the operations described herein which form part of the present invention; the operations are machine operations. Useful machines for performing the operations of the present invention include general purpose digital computers or other similar devices. In all cases the distinction between the method operations in operating a computer and the method of computation itself should be borne in mind. The present invention relates to method steps for operating a computer in processing electrical or other (e.g. mechanical, chemical) physical signals to generate other desired physical signals.

DETAILED DESCRIPTION OF THE INVENTION

1. General Solution to the Authorization Problem

The X Window System presently functions to allow access to host computers which may be connected to the server. The names of those host computers are stored in the server. By default, only programs on the local host computer and the host computers specified in an initial list can use the system. When a user attempts to use resources on the server, he need only use one of the host computers to be authorized to use the server and all of its resources. Moreover, a client on the local host (the host computer physically at the server) can change the access control list.

An X server can simultaneously use a number of different networking systems to communicate with the hosts running its clients. A client is a program running on a host computer. Each networking system will have its own way of running hosts, and X defines these ways as "address families."

A number of techniques have been developed to replace or augment this "host-based" authentication with a "user-based" authentication system which allows access to be restricted to individuals rather than to computers. One such technique is described below; it is based on the SunDES authentication mechanism used by the Sun Secure Remote Procedure Call package marketed by Sun Microsystems, Mountain View, California.

This mechanism provides a new "address family" in the X Window System which is called NetName. The NetName for a particular user is the

name by which that user is addressed within the X Window System. The term "user" as employed in this specification means a person having authorization to use the host computer. Since NetName is arranged as a new address family, it, like other address families, may be called by the X Windows ChangeHosts command to add and remove particular Net Names from the authorized list and by the ListHosts command to examine the names on the list. By providing NetName as a new address family, it is not necessary to add additional commands to the X Window System to access NetNames.

For each user of X Windows, a network-wide data base holds the NetName of the user, a public key, and a secret key encrypted with the user's password. It should be made clear at this point that many individuals may make up a single user under the same NetName if is it desirable to have them all working on a single project. Alternatively, a single person might have multiple NetNames representing different levels of security classification. A public key is an encryption coding device by which persons wishing to send encrypted messages to a single individual use that individual's public key to encrypt the message. The public key is, in effect, a code which an individual may give out to persons he wishes to address messages to him so that they may encrypt the messages for his receipt only. A secret key is a key used by the individual to decrypt those messages directed to him. Again, a secret key is a coded sequence which allows that individual only to decode the message addressed to him and encrypted by others by means of the public key.

When a user logs onto a host, the user password is used to decrypt his secret key and store it on the host computer. A process (called here a client) that wishes to contact the server constructs a credential encrypted with the

server's public key and containing the NetName of the client and a verifier including a time stamp to prevent replays of the particular message. The verifier is encrypted using the client's secret key. When the server receives the credential, the server uses its secret key to decrypt the credential. It then uses the client's public key to decrypt the verifier. Presuming the verifier can be decrypted by the client's public key, the server knows that only the user could have generated that credential. The credential is then checked against the network-wide database to determine the authorization of the particular user.

The manner in which the X Window System is changed in order to accomplish the above-described authorization is as follows. First the interface in the server to the authorization mechanism is altered to allow for multiple authorization mechanisms, and the host-based mechanism code is recast to fit into this new framework. The client X routine _XSendClientPrefix is altered to generate a suitable credential and transmit it to the server. To generate the credential, the routine needs (a) the client's NetName and (b) the server's NetName. The client's NetName is obtained from the client's operating system, and the server's NetName from an environment variable XNETNAME. If the environment variable XNETNAME is supplied, then a particular user is designated. If a name is not supplied in the environment position for the routine, the routine assumes that the NetName of the user running the server is the same as that of the user running the client.

An authorization mechanism is implemented in the server, alongside the host-based mechanism, that takes the credential from a client, decrypts it to obtain the NetName of the user running the client, and compares the NetName

with the net names in the authorized list. If there is no comparison, the connection is refused.

This authorization mechanism also maintains the list of authorized net names. It adds and deletes entries when a ChangeHost request with the family name NetName is encountered. The authorization mechanism also returns the entries from the list on a ListHost request. The routine clients/xhost/xhost.c is also modified to enable it to understand the NetName address family. Furthermore, the files clients/xdm/verify.c and clients/xdm/session.c are modified to set up the X NetName environment variable for the session to provide the NetName of the user running the server, to set up the initial authorized NetName list of the server so that the user who logged in is allowed to talk to the server, and to inform the host that the user has successfully logged on.

Included in Appendix A to this specification is the C language source code utilized to accomplish the foregoing limitation of access to only authorized users. A description of the steps for accomplishing authorization is illustrated in Figure 1.

2. Access Control

In order to prevent different users sharing an X server from interfering with or observing each other, the system of this invention provides the set of clients being run on behalf of each new user (more precisely, the set of clients authenticated under a single NetName or other authenticator) the illusion that they are the only clients using that server. In other words, clients under

authenticator A can neither observe nor affect the operation of clients under authenticator B.

To provide this illusion, the system of this invention insures that attempts by a client to name resources in the server, such as windows, created by some other client under a different authenticator will behave as if the resources had never been created..Further, attempts by a client to enquire about the existence and attributes of resources in the server will never return the names of resources created by clients under other authenticators. Since the client cannot discover the names of the resources, nor operate on them even if it could discover their names, the necessary isolation between authenticators is achieved.

This is accomplished in the present invention in the following manner. Clients (application programs) operate on resources in the X server by specifying a resource ID for them. Resource IDs form a flat 29-bit name space. Some resources, and the corresponding IDs, are generated by the server itself, and others by a client. At connection startup, each client is given a part of the resource ID space that it can use to generate IDs; it can access resources with IDs outside this space, but attempts to create resources with IDs outside its allocated space will fail with an IDChoice error. Since there are many more bits provided in the 29-bit name space than are necessary for client identification, some of these bits are utilized to provide user identification.

Figure 1 illustrates illustrates the manner is which access to the X Window server is provided in the standard arrangement of the prior art and in a server operating in accordance with the present invention. The upper diagram

of Figure 1 illustrates that in the prior art arrangement space is allocated to the server and to each individual client by means of the 29-bit name space. Any user may then access any of the clients. By allocating some of the most significant bits to the user identification as shown in the illustration of the second diagram in Figure 2,, the division of space is broken into much larger blocks each of which includes space for the client programs which can be viewed only by a single user (those using a single authenticator). In the illustration of the second diagram in Figure 2, the first block is allocated to the server and to clients of a window manager, a so-called privileged user. The next block of space is allocated to a user one for its clients and the next block to a user two and its clients.

The system ensures that attempts by clients to name (and thereby operate on) resources that were created by a client authorized under some other authenticator fail, just as they would have had the other client never created the resource. X Windows provides no resource ID for a client as such, only for resources created by the client. Since a client cannot name resources created by clients with other authenticators, it has no idea that these other clients exist. Thus, user one is only able to operate upon those clients in its user space. In like manner, user two is only able to operate upon those clients in its user space.

The problem is not as straightforward as it may seem. A client has to be able to name some resources not created by clients under its authenticator. In particular, normal clients must be able to name resources created by the server, such as the root window of the display, in order to operate normally; and some privileged clients must be able to name and operate on all resources.

Without this, a system would require that each user accessing an X server run a separate window manager to manipulate only that user's windows.

To deal with this, the invention introduces the concept of authenticator zero; clients under authenticator zero can name all resources irrespective of who created them, and all clients can name resources created by clients under authenticator zero. In other words, to clients under authenticator zero the X server behaves as it would normally, there are no restrictions. Authenticator zero refers to the user running the server itself, thus resources created by the server may be named by all clients. Normally, the window manager and a special selection agent are the only actual clients run under authenticator zero. The space allocated to the server and to authenticator zero is the space to the far left in the diagram on the second line of Figure 1.

Thus, the actual goal of the invention is to provide each set of clients authorized under a single non-zero authenticator (normal clients) the illusion that they, and the clients authorized under authenticator zero (privileged clients), are the only clients connected to the server. However, this is not in itself sufficient to ensure that the system is reasonably secure against malicious clients. As pointed out, normal clients must not be able to observe the operations of other normal clients, for example by snooping on keystrokes directed to them. Moreover at least one additional restriction is required; normal clients must not be able to destroy or modify a privileged client's resources.

The implementation of this design involves changing the routine that approves each authenticator to return an unique 32-bit value characteristic of

each valid authenticator. AddHost and RemoveHost maintain an array of up to thirty-two such 32-bit values. Each client is labelled with the index in this array of its authenticator. The interface to LookupID() is changed to include a client pointer. Each time a resource ID is looked up, the bits making up the user identification are ORed into the ID. This leaves twenty-four of the twenty-nine bits for use as a normal ID, enough for sixty-four simultaneous clients with each authenticator. Before calling the FreeResource routine, the extra five bits are masked off.

Thus, a system, X Windows, which is inherently non-secure to a user authenticated to use the system is rendered substantially more secure through the method of precluding that user from knowing that processes and users for which he is not authenticated exist. A user coming online and providing the correct NetName user identification and secret password sees only processes which he is allowed to see and is able to obtain the names of other processes and users only if they are within in the ambient of his authorization.

The implementation and design of this invention involves changing a number of routines and providing a list of authenticated codes for each individual client.

Specific tests were added to the following routines in order to accomplish the isolation of each individual user to its authenticated area:

CreateWindow and ReparentWindow -- the parent must be able to name the child and vice versa.

QueryTree -- don't return any children that the caller cannot name.

GetSelectionOwner -- return None if the caller cannot name the owner.

ConvertSelection -- refuse the conversion if the requestor cannot name the owner.

TranslateCoordinates -- return None if the caller cannot name the child.

ListInstalledColormaps -- do not include any unnameable colormaps in the returned list.

GetInputFocus -- return None if the caller cannot name the focus window.

QueryPointer -- return None if the caller cannot name the child window.

ChangeWindowAttributes and GetWindowAttributes -- ensure that the window can name the colormap.

Configure Window -- ensure that the above Sibling is a window that the client can name.*

In addition, a normal client cannot change or in any way interfere with the resources created by a normal client under some other authenticator because it cannot name them. But it can name the resources created by the server itself, and by privileged clients. To prevent malicious clients interfering with the work of the privileged clients, changes are needed to prevent a normal client changing or destroying privileged client's resources.

Normal clients could use the DestroyWindow, DestroySubwindows, MapWindow, MapSubwindows, UnmapWindow, UnmapSubwindows, ConfigureWindow, ReparentWindow, ChangeWindowAttributes, ChangeSaveSet and FreePixmap requests to interfere with privileged clients,such as those used by the window manager. Changing these requests to fail if the client is unprivileged and if it did not create the windows causes no problems for normal clients. Similarly, normal clients could interfere with the appearance of a privileged client by drawing in its windows and pixmaps. Changing the drawing requests to fail if the client is unprivileged and the drawable was created by a privileged client causes no problems for normal clients.

Normal clients could interfere with privileged client's cursors by using the FreeCursor and RecolorCursor requests. Changing these processes to fail if the client is unprivileged and the cursor was created by a privileged client causes no problems for normal clients.

Normal clients could free the fonts being used by a privileged client. Changing the FreeFont request to fail if the client is unprivileged and the font was opened by a privileged client cures this possible problem.

Normal clients could free a graphics context being used by a privileged client. Changing the FreeFont request to fail if the graphics context was not created by the requesting client will cause no problems; graphics contexts should not be shared between clients at all.

Normal clients could free a colormap being used by a privileged client. Changing FreeColormap to fail if the client is unprivileged and the colormap was created by a privileged client should not cause any problems; only privileged clients should be destroying shared colormaps.

Normal clients could use ChangeProperty, DeleteProperty or RotateProperties to alter properties on windows belonging to privileged clients. Restricting normal clients to using these requests on their own windows will not cause problems. According to the Inter-Client Communication Conventions manual, a normal client requesting the conversion of a selection has to supply one of its own windows. The owner (in this case the privileged selection agent) has to store the reply properties on this window. There is thus no reason for normal clients to be storing or changing properties on windows they don't own.

Included in Appendix B to this specification is the C language source code utilized to accomplish the foregoing limitations of users who have access to the system. The steps for accomplishing access control are illustrated broadly in Figure 2.

Although various methods have been described for making more secure the operation of a server system running the X Window System, various other extensions of the present invention and variations therein will be obvious to those skilled in the art without departing from the spirit and scope of the invention. The invention should therefore be measured by the claims which follow.

CLAIMS

1. The method of rendering an X Window System including a server and at least one client process secure comprising the steps of allowing users to view only resources of the X Windows server system the use of which has been specifically authorized to that user, and allowing users to manipulate only resources of the X Windows server system the use of which has been specifically authorized to that user.

2. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 1 in which the step of allowing users to view only resources of the X Windows server system the use of which has been specifically authorized to that user includes the steps of placing a user identification in an identification of each resource created by the user, and allowing the user to view resources created by that user and by an authenticator zero.

3. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 1 in which the step of allowing users to manipulate only resources of the X Windows server system the use of which has been specifically authorized to that user includes the step of allowing a user to manipulate only resources containing resource identifications containing a user identification indicating that user or indicating an authenticator zero.

4. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 1 in which the step of allowing users to view only resources of the X Windows server system the use of which has been specifically authorized to that user includes the steps of placing a user identification in an identification of each resource created by the user, and allowing the user to view resources created by that user and by an authenticator zero; and the step of allowing users to manipulate only resources of the X Windows server system the use of which has been specifically authorized to that user includes the step of allowing a user to manipulate only resources containing resource identifications containing a user identification indicating that user or indicating an authenticator zero..

5. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 1 further comprising the step of restricting access to the system to authorized users at host computer terminals.

6. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 5 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the X Windows server system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the client's secret key; decrypting the credential at the X Windows server using the

X Windows server's secret key to obtain the verifier; and decrypting the verifying to obtain the NetName of the client; and comparing the NetName of the client with the address family recognizable by the X Windows server system as containing authorized names for users to determine whether the user should be granted access to the X Window System.

7. The method of rendering an X Window System including a server and at least one client process secure comprising the steps of allowing client processes to view only resources of the X Window System the use of which by the user on whose behalf the client processes are operating has been specifically authorized, and allowing users manipulate only resources of the X Windows server System the use of which by the user on whose behalf the client processes are operating has been specifically authorized.

8. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 7 further comprising the step of restricting access to the server to authorized users.

9. The method of rendering an X Window System including a server and at least one client process secure as claimed in Claim 8 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the X Windows server system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the

client's secret key; decrypting the credential at the X Windows server using the X Windows server's secret key to obtain the verifier; and decrypting the verifying to obtain the NetName of the client; and comparing the NetName of the client with the address family recognizable by the X Windows server system as containing authorized names for users to determine whether the user should be granted access to the X Window System.

10. The method of rendering an X Windows server system running on a server and at least one host computer terminal secure comprising the steps of restricting access to the system to authorized users, allowing authorized users to view only resources of the X Window System the use of which has been specifically authorized, and allowing authorized users to manipulate only resources of the X Window System the use of which has been specifically authorized.

11. The method of rendering an X Windows server system running on a server and at least one host computer terminal secure as claimed in Claim 10 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the X Windows server system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the client's secret key; decrypting the credential at the X Windows server using the X Windows server's secret key to obtain the verifier; and decrypting the verifying to obtain the NetName of the client; and comparing

the NetName of the client with the address family recognizable by the X Windows server system as containing authorized names for users to determine whether the user should be granted access to the X Windows server system.

12. The method of rendering a networked window system running on a server and at least one host computer terminal secure comprising the steps of allowing users to view only resources of the networked window system the use of which has been specifically authorized to that user, and allowing users to manipulate only resources of the networked window system the use of which has been specifically authorized to that user.

13. The method of rendering a networked window system running on a server and at least one host computer terminal secure as claimed in Claim 12 further comprising the step of restricting access to the system to authorized users at host computer terminals.

14. The method of rendering a networked window system running on a server and at least one host computer terminal secure as claimed in Claim 13 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the networked window system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the client's secret key; decrypting the credential at the server using the server's secret key to obtain the verifier; and decrypting the verifying

to obtain the NetName of the client; and comparing the NetName of the client with the address family recognizable by the networked window system as containing authorized names for users to determine whether the user should be granted access to the networked window system.

15. The method of rendering a networked window system running on a server and at least one host computer terminal secure comprising the steps of allowing users to view and manipulate only resources of the networked window systems the use of which by that user has been specifically authorized, and restricting access to the server to authorized users.

16. The method of rendering a networked window system running on a server and at least one host computer terminal secure as claimed in Claim 15 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the networked window system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the client's secret key; decrypting the credential at the server using the server's secret key to obtain the verifier; and decrypting the verifying to obtain the NetName of the client; and comparing the NetName of the client with the address family recognizable by the server system as containing authorized names for users to determine whether the user should be granted access to the networked window system.

17. The method of rendering a networked window system running on a server and at least one host computer terminal secure comprising the steps of restricting access to the system to authorized users, allowing authorized users to view only resources of the networked window systems the use of which has been specifically authorized, and allowing authorized users to manipulate only resources of the networked window systems the use of which has been specifically authorized.

18. The method of rendering a networked window system running on a server and at least one host computer terminal secure as claimed in Claim 17 in which the step of restricting access to the system to authorized users at host computer terminals comprises the steps of providing an address family recognizable by the networked window system as containing authorized names for users, storing authorized names for users in a network-wide server database with a public key and a secret key known only to the user, generating a credential at a host computer terminal encrypted by the host computer terminal public key, the credential containing the NetName of the client and a verifier including the client's secret key; decrypting the credential at the networked window using the server's secret key to obtain the verifier; and decrypting the verifying to obtain the NetName of the client; and comparing the NetName of the client with the address family recognizable by the networked window system as containing authorized names for users to determine whether the user should be granted access to the networked window system.

19. The method of rendering an X Window System secure, substantially as hereinbefore described with reference to the accompanying drawings.